Today: SPACE!

- L, NL = coNL
- Log-space reductions
- NL-completeness
- Review Savitch's Thm

11/18/05 (1)

Q] Show CYCLE = {⟨G⟩ | G is a directed graph which contains a cycle}

is NL-Complete.

Proof] First we argue that CYCLE ∈ NL

Algorithm:

1. Nondeterministically guess start node s
2. Set counter ← 0, current-node ← s
3. Repeat until s is reached again, or counter = n:
   i) current-node ← Nondeterministically guess a neighbor of current-node. If it is s, <u>accept</u>.
   ii) counter ← counter + 1
4. If s was never found, reject

Analysis: Log n space to remember s, counter, current-node

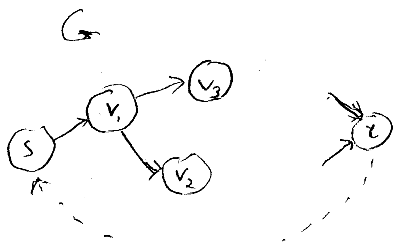
So CYCLE ∈ NL

Now must show PATH $\leq_L$ CYCLE

must give f s.t. f(⟨G, s, t⟩) = ⟨G'⟩

where G' has cycle iff G has path from s to t

and f uses only log workspace

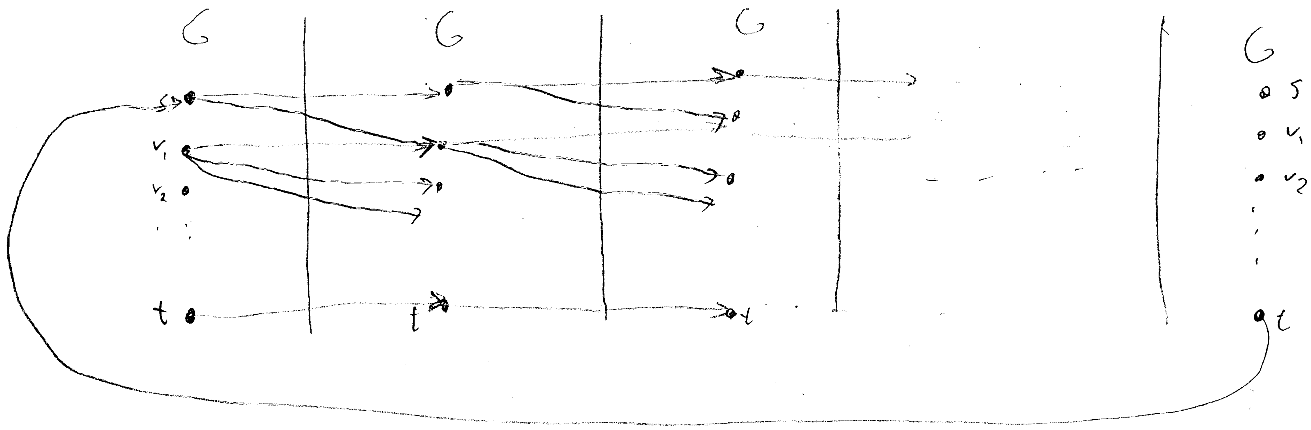


First crack: add edge from t → s

Now if G has path from s to t, G' has cycle s → t → s

But G' may have cycle even if G has no s → t path

How to fix this? with a <u>layered</u> graph G'



Can we compute this G' with a log-space transducer? Yes

input: $v_1, v_2, \ldots, v_n$, $(v_1, v_3), (v_2, v_3)$ ..... s, t

output: $v_1^1, v_1^2 \cdots v_1^n$, $v_2^1, v_2^2 \cdots v_2^n$, ... (v_1^1, v_3^2) (v_1^2, v_3^3) (v_1^3, v_3^4) ....

Transducer only needs $O(\log n)$ workspace to remember n, and some counters

snippet of how transducer works:

1) Count # nodes, store value n
2) For $i=1,2 \cdots n$, $j=1,2 \cdots n$
   Print v_i^j
3) For every input edge (v_m, v_n):
   For $i = 1, 2, \ldots n-1$, Print (v_m^i, v_n^{i+1})

Q] Is $\overline{\text{CYCLE}}$ NL-complete?

A] Yes. Note $\overline{\text{PATH}} \leq_L \text{PATH}$ (since PATH is NL-complete, and $\overline{\text{PATH}} \in \text{NL}$)

So $\text{PATH} \leq_L \overline{\text{PATH}}$

And $\overline{\text{PATH}} \leq \overline{\text{CYCLE}}$ (since we showed $\text{PATH} \leq \text{CYCLE}$)

So $\text{PATH} \leq_L \overline{\text{CYCLE}}$

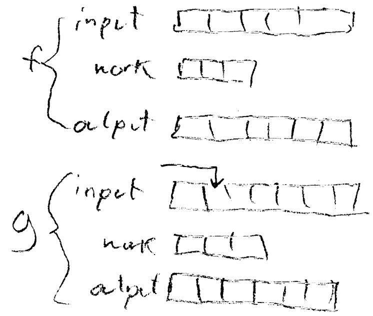
$\Downarrow$

$\overline{\text{CYCLE}}$ is NL-complete.

Note: same arg shows that complement of any NL-complete Lang is NL-complete

Note if $A \leq_L B$ and $B \leq_L C$ then $A \leq_L C$

Proof: Let f and g



Log-space transducer for $g \circ f$:

Run g, when g's READ head is on cell i, run f until it outputs i^{th} bit (ignore rest of f's output)
When g moves its READ head to cell $i+1$ (or $i-1$), re-run f to compute bit $i+1$ (or $i-1$).

Review of Space Classes:

$L \subseteq NL \subseteq P$

Since PATH is in P, and all languages in NL are log-space (hence poly-time) reducible to PATH

$L \subseteq NL \subseteq SPACE(\log^2 n)$ (sometimes written L^2)

Since Savitch's Thm yields a $O(\log^2 n)$ space algorithm for PATH

$O(\log^2 n)$ space algorithm for PATH:

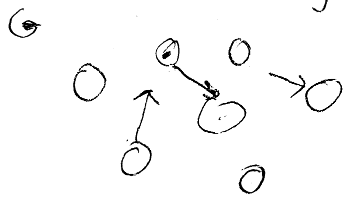
Run recursive algorithm K-STEP-PATH (G, a, b, K) outputs YES if there is a path from $a \rightsquigarrow b$ in K or fewer steps, NO otherwise

K-STEP-PATH(G, a, b, K)

1) If $K < 0$, return NO
2) If $a = b$, return YES
3) For all $c \in V$
   - If K-STEP-PATH$(G, a, c, K/2)$ and K-STEP-PATH$(G, c, b, K/2)$ both return YES, then return YES
4) Return NO (nothing in step 3 worked)

Analysis: Recursion depth is $O(\log n)$. At each level, must remember constant number of nodes, each with a $O(\log n)$ size pointer. Total is $O(\log^2 n)$.

Can think of algorithm as using $O(\log n)$ pointers, each of $O(\log n)$ size at any given time.



Note: Alg uses $O(\log^2 n)$ space but is not polynomial time

Aside: Undirected PATH problem can be solved in log space (and hence polynomial time simultaneously) due to a very recent result of Omer Reingold last year.